**MODEL SUMMARY**

**The image classification task is addressed using convolutional Neural Network, chosen for its versatality in applications such as identifying satellite imaginery, analysing medical images,forcasting time series etc.**

**Convolution Layers:**

* **Core building block of CNN.**
* **Comprises few components, including input data, filter, and feature map.**
* **Feature Detector (kernel/filter) moves across the image's receptive field, performing convolution.**
* **Convolution involves applying the filter to an image area, calculating dot products, and shifting by a specified stride until the entire image is covered.**
* **Activation Functions:**
  1. **Essential component in CNN.**
  2. **Introduces non-linearity to the model.**
  3. **Enables the network to learn complex patterns and relationships in data.**
* **Pooling Layer:**
  1. **Also known as down-sampling layers.**
  2. **Purpose is to decrease the number of parameters in input data.**
  3. **Two primary types:**
     + **Max pooling: Selects the maximum value within a pixel set and forwards it to the output array.**
     + **Average pooling: Computes the average value within the receptive field and forwards it to the output array.**
* **Convolutional Neural Networks (CNNs), functioning as feedforward neural networks, demonstrate proficiency in analyzing visual images through the processing of data organized in a grid-like structure.**
* **Within the convolutional layer, diverse filters perform convolution operations, treating each image as a matrix of pixel values.**
* **The filter matrix slides across the image matrix, conducting dot product computations at each step, ultimately producing a convolution feature matrix.**
* **Image Processing:**

**Employ OpenCV to load each image.**

**Convert images to the RGB color space.**

**Resize images uniformly to (128, 128).**

**Convert the processed images into NumPy arrays.**

* **Model Architecture:**

**Utilize a sequential CNN model consisting of:**

**Convolutional layer with 32 filters, a 3x3 kernel size, and ReLU activation.**

**Max-pooling layer with a 2x2 pool size.**

**Flatten layer to transform the 2D feature map into a 1D vector.**

**Dense layer with 256 units and ReLU activation.**

**Dropout layer with a 0.1 dropout rate to prevent overfitting.**

**Another dense layer with 512 units and ReLU activation.**

**Output layer with 5 units using softmax activation for multi-class classification.**

* **Model Compilation:**

**Compile the model using Adam optimizer.**

**Utilize sparse categorical cross-entropy loss, with accuracy as the metric.**

* **Model Training:**

**Split the dataset for training, allocating 70% for training and 30% for testing (using train\_test\_split from scikit-learn).**

**Normalize both training and testing data using TensorFlow to scale pixel values between 0 and 1.**

**Train the model for approximately 200 epochs with a 0.1 validation split.**

* **Key Observations:**
  + **Achieving a commendable 84% accuracy after 200 epochs is notable.**
  + **However, the model's effectiveness appears influenced by the constrained dataset size, consisting of only 150 images.**
  + **The limited dataset poses challenges for the model to capture comprehensive patterns, leading to a potential inclination towards overfitting.**
* **Improvement Strategies:**
  + **To enhance the model's performance:**
    - **Consider expanding and diversifying the training dataset.**
    - **Experiment with diverse regularization methods to mitigate overfitting tendencies.**